NL0100065
6 febr.

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :
G06F 9/32, 9/46

A1

(11) International Publication Number: **WO 99/18497**

(43) International Publication Date: 15 April 1999 (15.04.99)

(21) International Application Number: PCT/SE98/01699

(22) International Filing Date: 22 September 1998 (22.09.98)

(30) Priority Data:
08/946,639 7 October 1997 (07.10.97) US

(71) Applicant: TELEFONAKTIEBOLAGET LM ERICSSON (publ) [SE/SE]; S-126 25 Stockholm (SE).

(72) Inventors: HAMMAR, Claes, Lennart; Travarevägen 14, S-187 54 Täby (SE). HÅKANSSON, Stefan; Sandviksgatan 44B, S-972 32 Luleå (SE).

(74) Agent: ERICSSON RADIO SYSTEMS AB; Common Patent Dept., S-164 80 Stockholm (SE).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: METHOD FOR IMPROVED INTERRUPT HANDLING WITHIN A MICROPROCESSOR

(57) Abstract

A method and apparatus for processing interrupts (30) received by a processor (45) during the processing of an instruction set by a processing pipeline (55) is disclosed. Initially, an interrupt (30) associated with an interrupt service routine is received during processing of the instruction set. In response to receipt of the interrupt (30), an interrupt instruction register (50) is accessed containing at least one interrupt instruction associated with the interrupt service routine. The at least one interrupt instruction is inserted directly into the processing pipeline (55) of the processor (45). The inserted instruction is then executed and control returns back to processing of the original instruction set after execution of at least one interrupt instruction.

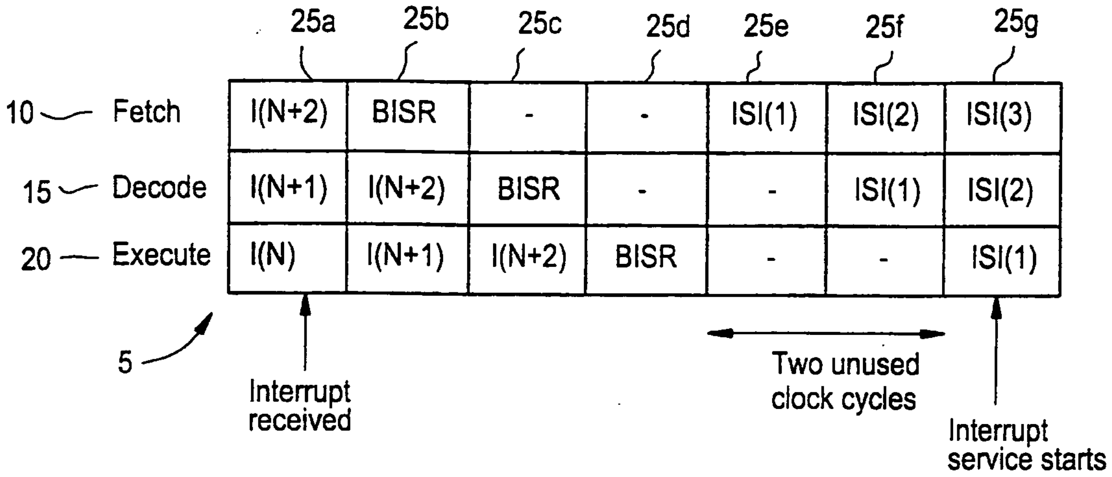
- het ~~voornaamste~~ verschil is dat wy het (ook) voor een function call doen (dus niet alleen voor een interrupt)
- en dat wy na de "ingevoegde" instructies de rest van de call afhandelen

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

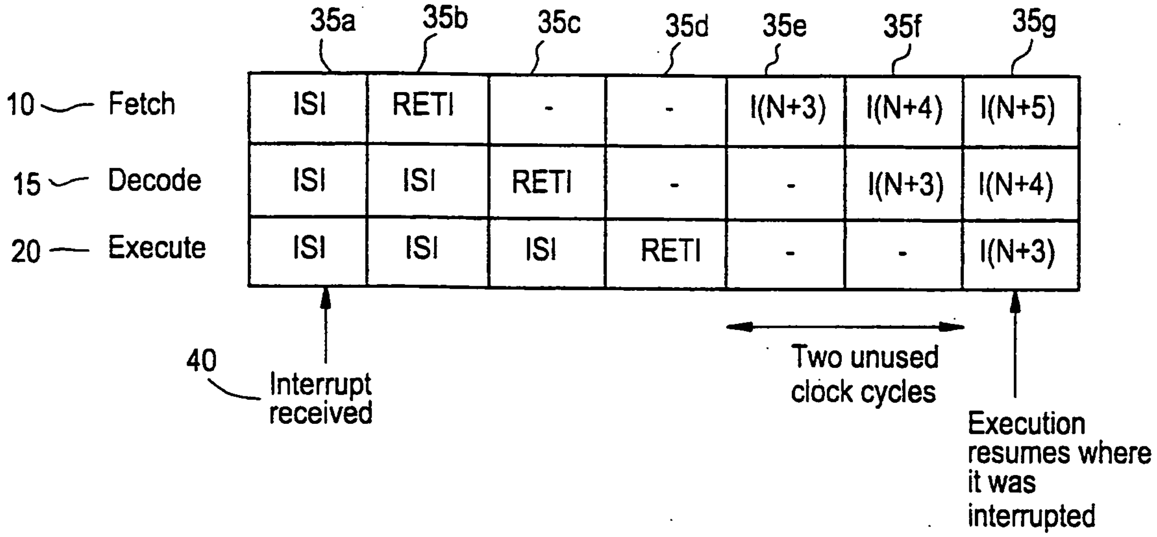
AL	Albania
AM	Armenia
AT	Austria
AU	Australia
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CU	Cuba
CZ	Czech Republic
DE	Germany
DK	Denmark
EE	Estonia
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GH	Ghana
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IL	Israel
IS	Iceland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakstan
LC	Saint Lucia
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LS	Lesotho
LT	Lithuania
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
MK	The former Yugoslav Republic of Macedonia
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
MX	Mexico
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SG	Singapore
SI	Slovenia
SK	Slovakia
SN	Senegal
SZ	Swaziland
TD	Chad
TG	Togo
TJ	Tajikistan
TM	Turkmenistan
TR	Turkey
TT	Trinidad and Tobago
UA	Ukraine
UG	Uganda
US	United States of America
UZ	Uzbekistan
VN	Viet Nam
YU	Yugoslavia
ZW	Zimbabwe

FIG.1



		25a	25b	25c	25d	25e	25f	25g
10	Fetch	I(N+2)	BISR	-	-	ISI(1)	ISI(2)	ISI(3)
15	Decode	I(N+1)	I(N+2)	BISR	-	-	ISI(1)	ISI(2)
20	Execute	I(N)	I(N+1)	I(N+2)	BISR	-	-	ISI(1)

FIG.2



		35a	35b	35c	35d	35e	35f	35g
10	Fetch	ISI	RETI	-	-	I(N+3)	I(N+4)	I(N+5)
15	Decode	ISI	ISI	RETI	-	-	I(N+3)	I(N+4)
20	Execute	ISI	ISI	ISI	RETI	-	-	I(N+3)

FIG.3

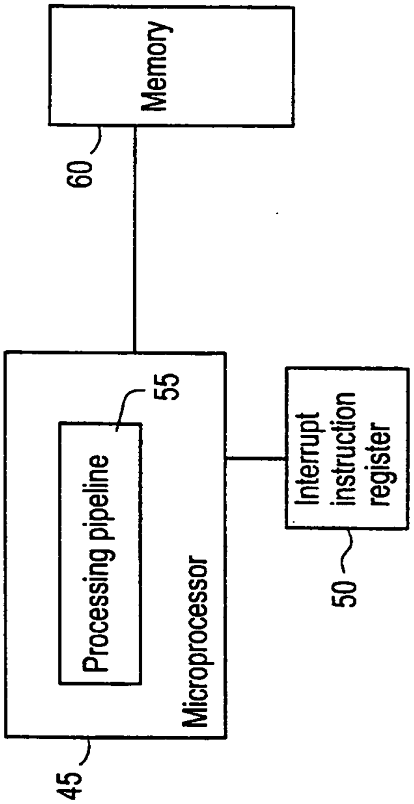
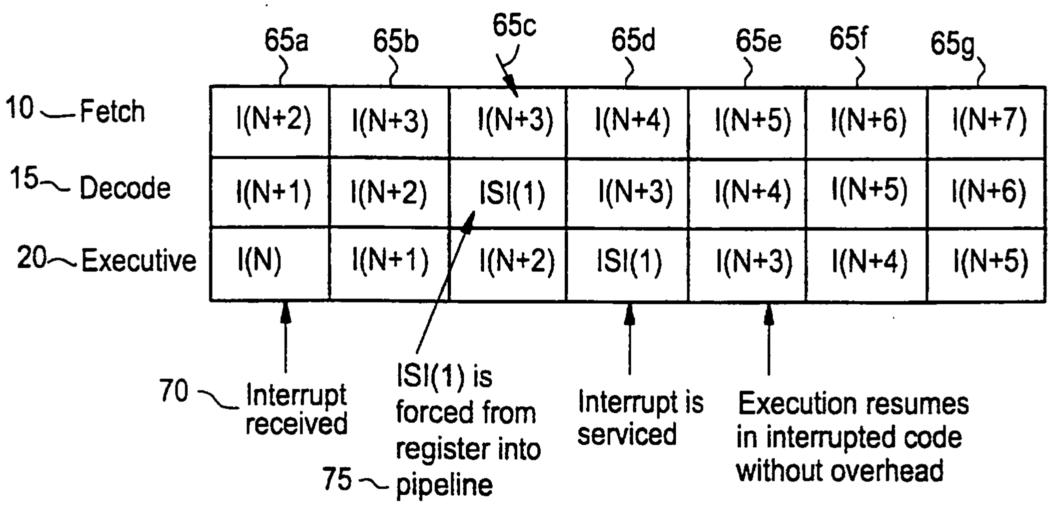

Memory
60
Processing pipeline
55
Microprocessor
45
Interrupt instruction register
50

FIG.4



	65a	65b	65c	65d	65e	65f	65g
10 Fetch	I(N+2)	I(N+3)	I(N+3)	I(N+4)	I(N+5)	I(N+6)	I(N+7)
15 Decode	I(N+1)	I(N+2)	ISI(1)	I(N+3)	I(N+4)	I(N+5)	I(N+6)
20 Executive	I(N)	I(N+1)	I(N+2)	ISI(1)	I(N+3)	I(N+4)	I(N+5)

70 Interrupt received
ISI(1) is forced from register into
75 pipeline
Interrupt is serviced
Execution resumes in interrupted code without overhead

FIG.5A

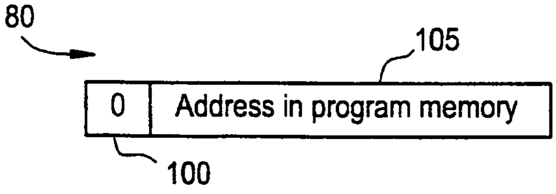
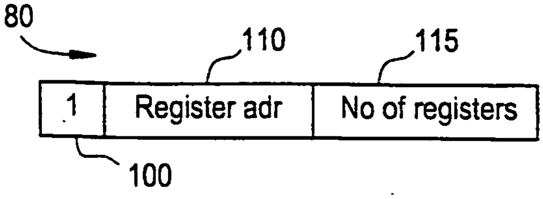


FIG.5B



METHOD FOR IMPROVED INTERRUPT HANDLING WITHIN A MICROPROCESSOR

BACKGROUND OF THE INVENTION

Technical Field of the Invention

The present invention relates to instruction processing by a microprocessor, and more particularly, to the processing of interrupts by a microprocessor.

Description of Related Art

Within most microprocessors, such as micro-controllers, digital signal processors, embedded CPUs, etc., the processing of a current task (instruction set) may be stopped to enable the execution of another task related to the current task in response to some external event. The external event is commonly called an interrupt to the microprocessor. The interrupt causes the microprocessor to branch to and begin processing an interrupt service routine consisting of a number of interrupt instructions. After completion of processing of the interrupt service routine, the processor switches back to the original task and resumes processing at the point that it was interrupted.

Processors usually employ a so called pipeline structure for the execution of an instruction set defining a task. A pipeline structure refers to the division of the execution of each instruction into several stages. These stages normally comprise a fetch, decode and execute stage. The fetch stage involves retrieving the instruction from memory for execution. The decode stage involves interpreting the instruction to decide what actions are to be taken, and the execution stage involves actually executing the determined actions.

Currently used methods for processing interrupts to a processor involve branching to a location in memory where the interrupt service routine is stored. Once execution of the interrupt service routine has been completed, a branch returns processing back to where the interrupt stopped processing on the previous task. Unfortunately, the process of branching to the set of instructions defining the interrupt service routine requires the use of overhead to process the interrupt. Overhead involves the execution of clock cycles by the processor where no instructions are actually executed. The branching process requires the execution of two unused clock cycles until the first instruction of the interrupt service routine is executed. Likewise, an overhead of two clock cycles is required to branch back from the interrupt service routine to the original instruction set. Thus, a total of four clock cycles would remain unused upon the execution of each interrupt service routine. If actual processing of the interrupt service routine required only one clock cycle (a single instruction) the overhead would be 400%. Thus, some means of more efficiently processing interrupt service routines that limits overhead generated by the procedure would greatly improve the utilization of processing resources.

Another disadvantage of current systems arises from the fixed location of the code needed to execute a specific interrupt service routine. If different actions must be taken for a specific interrupt service routine depending on when the interrupt is received, the program must introduce "if-then-else" portions within the interrupt service routine code or must change the contents of the program memory for the interrupt service routine to ensure a return to the correct location. The introduction of "if-then-else" portions wastes clock cycles and memory and is thus an unacceptable solution. The changing of the contents of the program memory is a risky and impracticable solution. The fixed locations in memory of the interrupt service routines also means that if one interrupt does not require all of the space reserved in memory for that interrupt, other interrupts may not utilize the unused memory. Thus, means for more efficiently utilizing the available memory resources within a processor with respect to interrupt service routines is also needed.

SUMMARY OF THE INVENTION

The present invention overcomes the foregoing and other problems with a system and method providing for improved processing of interrupts received by a processor during execution of an instruction set. Initially, while the processor is executing an instruction set through the processing pipeline, an interrupt is received that is associated with a particular interrupt service routine including at least one instruction.

In response to reception of the interrupt, the interrupt is examined to determine the contents of configuration fields within the interrupt. A first configuration field indicates whether the interrupt is to be processed using an interrupt instruction register or is to be processed according to a more conventional branch interrupt service procedure. A second configuration field denotes the address of the interrupt instruction register in the case of the interrupt to be processed by an interrupt instruction register, and a third configuration field indicates the number of instructions included within the interrupt service routine.

Using the information on the address of the interrupt instruction register, the identified register is accessed to extract the at least one interrupt instruction contained therein. This instruction is inserted directly into the processing pipeline of the processor. At the same time, the most recently fetched instruction within the processing pipeline is again fetched into or retained within the fetch stage of the processing pipeline. The inserted at least one interrupt instruction is executed, and the processor returns to processing the instruction set after execution of the at least one interrupt instruction.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention, reference is made to the following detailed description taken in conjunction with the accompanying drawings wherein:

FIGURE 1 is a diagram of a processing pipeline executing a branch to an interrupt service routine in accordance with the prior art;

FIGURE 2 is a diagram of a processing pipeline executing a return from an interrupt service routine in accordance with the prior art;

FIGURE 3 is a functional block diagram of a microprocessor system including interrupt instruction registers;

FIGURE 4 is a diagram of an instruction pipeline executing an interrupt service routine using interrupt instruction registers; and

FIGURES 5a and 5b are diagrams of configuration fields for interrupts initiating access to an interrupt service routine.

DETAILED DESCRIPTION OF THE INVENTION

Referring now to the Drawings, and more particularly to FIGURE 1, there is illustrated the processing, in accordance with the prior art, of instructions by a microprocessor upon receipt of an interrupt. The interrupt causes processing to branch to an interrupt service routine. The instructions are processed through a processing pipeline 5. The processing pipeline 5 consist of a fetch stage 10, decode stage 15 and execute stage 20. These stages are aligned along the vertical axis of the processing pipeline 5.

The horizontal blocks represent clock cycles 25 executed by the microprocessor. An interrupt 30 is received during clock cycle 25a, while instruction I(N) is being executed and instruction I(N+2) is within the fetch stage of the pipeline 5. The received interrupt causes the processing of a "branch to interrupt service routine" (BISR) instruction within the pipeline 5 at clock cycle 25b. The BISR instruction cycles through the fetch 10 and decode 15 stages during clock cycles 25b and 25c until the execute stage 20 is reached. It should be noted that no instructions are inserted into the pipeline 5 behind the BISR instruction. When the BISR instruction reaches the execute stage 20 of the pipeline 5, processing branches to the interrupt service routine identified by the BISR instruction. The branching process requires an overhead of two unused clock cycles, namely, cycles 25e and 25f where no instructions are being executed.

Referring now to FIGURE 2, it can be seen that similar overhead requirements are needed when the interrupt service routine branches back to the original instruction set. During clock cycle 35a, a return from interrupt (RETI) instruction 40 is received to branch the interrupt service routine back to normal instruction set processing. The RETI instruction 40 is processed through the fetch 10 and decode 15 stages until the execution stage 20 is reached at cycle 35d. As before, no instructions are loaded behind the RETI instruction 40. As the normal processing instructions are retrieved for further processing at cycles 35e and 35f, these two clock cycles remain unused due to pipelining before the next instruction I(N+3) code is executed at the execution stage 20. This type of interrupt service routine processing wastes the clock cycles in which no instructions are executed. This causes an inefficient use of system resources and a slowing down of processing times.

Referring now to FIGURE 3, there is illustrated a functional block diagram of a microprocessor 45 using interrupt instruction registers 50 in association with the processing pipeline 5 to improve interrupt processing by substantially eliminating overhead processing. While the following description is made with respect to the use of an interrupt register, it should be realized that a RAM, FLASH memory or like storage means may be used instead of a register. The microprocessor 45 utilizes interrupt instruction registers 50 which may insert instructions directly into the stages of the processing pipeline 5. This eliminates the need for the interrupt instructions to be fetched from memory 60, and enables processing to begin directly at the decode stage 15 (FIGURE 4).

Referring now to FIGURE 4, there is more fully illustrated the processing pipeline 5 operating in association with interrupt instruction register 50. The processing pipeline 5 consist of the fetch 10, decode 15 and execute 20 stages as described previously. Execution of instructions through the pipeline 5 occur in response to each clock cycle 65 of the processor 45. During clock cycle 65a, an interrupt 70 is received by the processor 45 during execution of instruction I(N). The interrupt 70 is processed during the next clock cycle 65b to determine the configuration fields of the interrupt 70, as will be more fully discussed in a moment, and instructions continue to flow through the processing pipeline 5 in a normal fashion.

During clock cycle 65c, the first interrupt service instruction ISI(1) 75 is forced into the processing pipeline 5. In response to this forcing of ISI(1) into the decode stage 15, previously fetched instruction I(N+2) is again fetched from memory 60. Alternatively, the presently contained instruction I(N+2) may just be retained at its present location in the processing pipeline. The inserted interrupt service instruction ISI(1) is processed through the processing pipeline 5 and executed at the execution stage 20 during clock cycle 65d. It should be noted that an additional interrupt service instruction ISI(2) could be inserted into the processing pipeline at clock cycle 65d as is necessary to carry out all instructions necessary to complete an interrupt servicing routine. Should this occur, the instruction currently within the fetch stage 10 is again fetched until all instructions for completing the interrupt service routine are inserted into the processing pipeline 5. Once all necessary interrupt servicing instructions have

I (N+3)

been inserted within the processing pipeline 5 and executed at the execution stage 20, execution of a next instruction I(N+3) from the original instruction set resumes at cycle 65e. It can be seen that the processing of interrupts in this manner enables the execution of the interrupt service routine without the use of any overhead.

The interrupts received by the processing pipeline 5 may be configured to instruct the microprocessor 45 to process the interrupt according to a conventional interrupt branch routine or using interrupt instruction registers 50 by including a configuration field 80 with each interrupt. Referring now to FIGURES 5a and 5b, there are illustrated various configurations fields 80 for interrupts. FIGURE 5a illustrate an interrupt configuration field 80 enabling processing of the interrupt according to a conventional interrupt branch routine.. The identification bit 100 identifies whether the interrupt shall be serviced in the traditional way using an interrupt service routine stored in memory 60, or if the interrupt shall be served using an interrupt instruction register 50. If the identification bit 100 identifies execution of an interrupt service routine as shown in FIGURE 5a, the memory address field 105 of the interrupt points to an address in memory 60 where the interrupt service routine begins. Alternatively, the second portion 105 may include part of an address to be combined with a page pointer.

Referring now to FIGURE 5b, if the identification bit 100 indicates the interrupt shall be serviced by the interrupt instruction register 50, register address field 110 identifies the location of the register containing the first instruction to be executed. The number of registers field 115 indicates how many instructions must be fetched from the interrupt instruction registers to service the interrupt service routine associated with the interrupt.

Using these configuration fields 80 several different interrupts can share an interrupt service routine or interrupt instruction register 50. By associating each interrupt with a particular configuration field, the interrupt can be associated with one or several of the interrupt instruction registers 50 or with a selectable area in memory 60. Thus, the operation of the interrupt can be changed simply by altering the configuration fields 80 associated with the interrupt. Likewise, a new set of instructions or number of instructions may be indicated by altering the configuration of the interrupt. Furthermore, since the position in a memory 60 for an interrupt service routine is programmable, the positions for the routines can be optimized, thereby saving memory space.

Although a preferred embodiment of the method and apparatus of the present invention has been illustrated in the accompanying Drawings and described in the foregoing Detailed Description, it is understood that the invention is not limited to the embodiment disclosed, but is capable of numerous rearrangements, modifications, and substitutions without departing from the spirit of the invention as set forth and defined by the following claims.

WHAT IS CLAIMED IS:

1. A method for processing interrupts received by a processor processing an instruction set within a processing pipeline, comprising the steps of:

receiving an interrupt associated with an interrupt service routine including at least one interrupt instruction;

accessing an interrupt instruction register containing the at least one interrupt instruction;

inserting the at least one interrupt instruction of the interrupt service routine directly into the processing pipeline of the processor;

executing the at least one interrupt instruction of the interrupt service routine; and

returning to processing of the instruction set after execution of the at least one interrupt instruction.

2. The method of claim 1 wherein the step of accessing further includes the steps of:

reading a configuration field associated with the interrupt; and

determining from a first portion of the configuration field if the interrupt is processed using the interrupt instruction register.

3. The method of claim 2 wherein the step of accessing further includes the steps of:

reading a configuration field associated with the interrupt; and

determining from a second portion of the configuration field an address of the interrupt instruction register.

4. The method of claim 2 further including the step of determining from a third portion of the configuration field a number of interrupt instructions included in the interrupt service routine.

5. The method of claim 1 further including the step of maintaining within the processing pipeline a last fetched instruction from the instruction set during insertion of the at least one interrupt instruction.

6. The method of claim 5 wherein the step of returning comprises processing the maintained instruction after execution of the at least one interrupt instruction.

7. A system for executing an instruction set, comprising:

a processing pipeline for fetching and executing the instruction set; and

an interrupt instruction register responsive to receipt of an interrupt for inserting at least one interrupt instruction directly into the processing pipeline.

8. The system of claim 7 further including means for checking a configuration field of the interrupt to determine if the interrupt is to be processed using the interrupt instruction register.

9. The system of claim 7 further including means for checking a configuration field of the interrupt to determine an address of the interrupt instruction register.

10. The system of claim 7 further including means for checking a configuration field of the interrupt to determine a number of interrupt instructions to be inserted into the processing pipeline.

11. The system of claim 7 further including a memory for storing a plurality of interrupt service routines accessible by the processing pipeline in response to receipt of an interrupt.

12. The system of claim 11 further including means for checking a selected portion of a configuration field of the interrupt to determine a location in memory for

an interrupt service routine, the selected portion selectively programmable to a plurality of memory locations.

13. The system of claim 7 further including interrupts having selectively configurable configuration fields, the configuration fields including a selected portion indicating a location in memory for an interrupt service routine associated with the interrupt, the selected portion selectively programmable to a plurality of memory locations.

14. The system of claim 13 wherein the selected portion of the configuration field may direct multiple interrupts to the same memory location.

INTERNATIONAL SEARCH REPORT

Intern al Application No
PCT/SE 98/01699

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F9/32 G06F9/46

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	WO 93 01563 A (SEIKO EPSON CORP) 21 January 1993 see page 64, line 14 - page 65, line 30 ---	1,2,5-8, 11
Y	KLOKER K L: "THE MOTOROLA DSP56000 DIGITAL SIGNAL PROCESSOR" IEEE MICRO, vol. 6, no. 6, 1 December 1986, pages 29-48, XP000211994 see page 37, right-hand column, paragraph 2 - page 39, right-hand column, paragraph 1 ---	1,2,5-8, 11
A	US 5 117 498 A (MILLER GARY L ET AL) 26 May 1992 see abstract --- -/--	4,10

☒ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search 17 November 1998	Date of mailing of the international search report 25/11/1998
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Daskalakis, T

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No

PCT/SE 98/01699

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9301563 A	21-01-1993	EP 0547248 A JP 6502034 T	23-06-1993 03-03-1994
US 5117498 A	26-05-1992	NONE	
US 5155853 A	13-10-1992	JP 2062547 C JP 2144630 A JP 7095277 B	24-06-1996 04-06-1990 11-10-1995

INTERNATIONAL SEARCH REPORT

Inter   ıal Application No
PCT/SE 98/01699

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 155 853 A (MITSUHIRA YUKO ET AL) 13 October 1992 see the whole document -----	1,7

1